Integrating dynamic resources in corporate semantic web: an approach to enterprise application integration using semantic web services

Moussa Lo, Fabien Gandon

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Integrating dynamic resources in corporate semantic web: an approach to enterprise application integration using semantic web services

Moussa LO, Fabien GANDON

N° 5663

Août 2005

THÈME SYM

ISSN 0249-6399

Integrating dynamic resources in corporate semantic web: an approach to enterprise application integration using semantic web services

Moussa LO[1], Fabien GANDON [2]

Thème SYM – Systèmes symboliques
Projet Acacia

Rapport de recherche n° 5663 – Août 2005 - 35 pages

Abstract:
We present our experiment in integrating semantic web services in the existing semantic web server architecture used by the ACACIA team to implement corporate memories. We rely on CORESE, a semantic web search engine, to provide web applications based on the semantic web services it can identify. Thus, CORESE is used as a semantic UDDI registry and allows us to automatically discover and invoke corporate applications wrapped into semantically annotated web services. Using rules and an extension to the existing semantic web service frameworks, we also demonstrate how (i) to compose the web services with queries on the knowledge stored in the corporate memory to automatically populate the service inputs and (ii) to turn CORESE into a composable service of the memory.

Keywords: Corporate semantic web services, enterprise application integration

Acknowledgments: we thank AUF for partially supporting this work through the post-doctoral fellowship held by Moussa LO.

[1] Laboratoire d'Analyse Numérique et d'Informatique, Université Gaston Berger – BP 234 Saint-Louis, Sénégal, lom@ugb.sn
[2] INRIA National Institute of Research in Computer Science of Sophia Antipolis, France, Fabien.Gandon@sophia.inria.fr

Unité de recherche INRIA Sophia-Antipolis
2004, route des Lucioles, 06902 Sophia-Antipolis (France)
Téléphone : +33 (0)4 92 38 77 77 – Télécopie : +33 (0)4 92 38 77 65

Intégration de ressources dynamiques dans un web sémantique d'entreprise : une approche d'intégration d'applications d'entreprise utilisant les services web sémantiques

Résumé: Nous présentons notre première expérience d'intégration de services web sémantiques dans l'architecture de serveur web sémantique qu l'équipe ACACIA utilise pour implémenter des mémoires d'entreprise. Nous utilisons CORESE, un moteur de recherche sémantique, pour fournir des applications web basées sur des services web sémantiques qu'il peut identifier. D'abord, CORESE est utilisé comme registre sémantique UDDI et permet la découverte automatique et l'invocation dynamique d'applications d'entreprise transformées en web services annotés. En utilisant des règles de production et une petite extension des infrastructures de services web sémantiques existants, nous démontrons aussi comment (i) composer les web services avec des requêtes sur la connaissance stockée dans la mémoire et (ii) transformer CORESE en service composable avec la mémoire.

Mots clés: services web sémantiques d'entreprise, intégration d'application d'entreprise

Remerciements: nous remercions l'AUF pour avoir partiellement financé le post-doc de Moussa LO au sein de l'équipe ACACIA .

Table of contents

1 Introduction

Until the end of the 90's, enterprise modeling has been mainly used as a tool for enterprise engineering. But the new trends and the shift in the market rules led enterprises to become aware of the value of their memory and of the fact that enterprise model has a role to play in knowledge management (KM) too. Just as data-integration problem can benefit from corporate-level models, technology and application integration problem can benefit from these same models. This was recognized by practitioners of Enterprise Application Integration but it requires a programming paradigm at a level of abstraction high enough to ease its implementation.

In the past, the ACACIA team experimented with agent-based architecture for distributed KM [18]. At that time semantic web frameworks had not yet met the web services frameworks and all our architectures being at the knowledge level, we relied on agent-based frameworks for their implementation. With the emergence of frameworks to semantically annotate web services [1] [34][54], a new paradigm can now be used to integrate enterprise applications in a model-based memory.

In this report we describe our first experiment in integrating semantic web services with CORESE the existing semantic web search engine we use to build corporate semantic webs.

The three first sections introduce the issues and the state of the art. In section 2 we briefly introduce the need to get unified and integrated access to corporate applications and services and to integrate them with the corporate memory; we describe the needs to take into account dynamic resources which led us to integrate web services into corporate semantic webs. In section 3 we recall the vision of a corporate semantic web as developed by the ACACIA team and we summarize our previous work on corporate semantic webs. Section 4 is a survey of existing semantic web service frameworks and a positioning of our work; in particular we give our position in the semantic web service stack.

Section 5 presents our current implementation embedded in the semantic web server architecture. We describe our architecture which relies on CORESE, a semantic web search engine, and provides automatic discovery and invocation of annotated web services. Using production rules we also demonstrate how to facilitate services composition.

Finally section 6 tackles the original issue of composing corporate web services with knowledge from the corporate memory. We explore two paths: using semantic types to attach queries to service inputs and turning CORESE into a composable service of the memory.

2 Motivating scenario: corporate application management

> *"Organizations that are able to integrate their applications and data sources have a distinct competitive advantage: strategic utilization of company data and technology for greater efficiency and profit. But IT managers attempting integration face daunting challenges — disparate legacy systems; a hodgepodge of hardware, operating systems, and networking technology; proprietary packaged applications; and more. Enterprise Application Integration (EAI) offers a solution to this increasingly urgent business need. It encompasses technologies that enable business processes and data to speak to one another across applications, integrating many individual systems into a seamless whole."* [27]

More and more often, the ACACIA team must face scenarios requiring not only knowledge access but also computation, decision, routing, transformation, etc. Until now, our corporate semantic webs focused on providing us with a unified and integrated access to a range of knowledge sources; but there is a growing demand to get the same facility to access corporate applications and services and to integrate both worlds.

Users expect IT managers to get very different computing systems (desktops, mobile phone, PDA, mainframes, etc.) to talk together and, even worse, to get the variety of applications that run on them to talk together. But what does it mean to talk together? Who talks to whom? What are the flows and processes? What are the purposes?

Users don't only want to get access to the needed pieces of information, they want this information in a format they are used to, with some certification of quality or of provenance, with appropriate tools to analyze it, modify it, etc.

Usage scenarios are moving from a unified access to information to a unified access to information and applications. Corporate memories not only include information mediums but more generally:

- *information storage services* including: information diffusion systems (digital libraries, mailing-lists, forums, blogs, etc.) and dedicated systems (corporate or public databases, ERP, data warehouse, etc.);
- *information creation services* including: sensors (e.g. location tracking, presence & availability), computation and inference systems (e.g. data analysis tools);
- *information flows management services* including: secured transport channels, business rule engines and workflow systems, connectivity management, privacy enforcement and trust propagation;
- *information mediation services* including: matchmaking directories, translation and mapping services, contract and service quality enforcement;
- *information presentation services* including: multimedia transformation and translation, contextual adaptation, dynamic customization and manipulation interfaces;

All these services may be internal or external to the company yet users want them to interoperate smoothly and, even better, to automatically integrate their workflows at the business layer.

3 Corporate semantic webs as document-based corporate memories

Semantically annotated information worlds are, in the actual state of the art, an effective way to make information systems smarter. If a corporate memory becomes an annotated world, corporate applications can use the semantics of the annotations and through inferences help the users use the corporate memory.

The ACACIA team at INRIA focuses on knowledge management solutions based on semantic Web technologies. As shown in Figure 1 from [18], we use RDF Model, RDF Schema and OWL (essentially OWL Lite) to describe ontologies and implement knowledge models. Organizational entities and people are annotated in RDF and its XML syntax is used to store and exchange the annotations. This choice enables us to base our system on the W3C recommendations that benefit from all the web-based technologies for networking, display and navigation. This clearly is an asset for the integration to a corporate intranet environment that often relies on web technologies. Relying on W3C standards also enables us to integrate access to external sources in the corporate memory (e.g. digital libraries offering references in the application domain), interconnect parts of intranets to form extranets, generate focused portals for customized access (e.g. to address device independence, mobile access, etc.), etc. Clearly relying on open standards is important for effective knowledge representation and knowledge management solutions.

This work resulted in the development of a semantic Web search engine (CORESE [9]) enabling us to analyze, query and infer from descriptions in RDF(S)/OWL. CORESE implements a query language close to SPARQL [44] and a production rule language used to declare domain-dependent inference rules. CORESE was tested with a variety of schemas such as the Gene ontology (13700 concept types). It also provides approximate search capabilities (vital to information retrieval systems) and comes with a semantic web server providing the user with presentation capabilities to dynamically generate query interfaces and templates to render results.

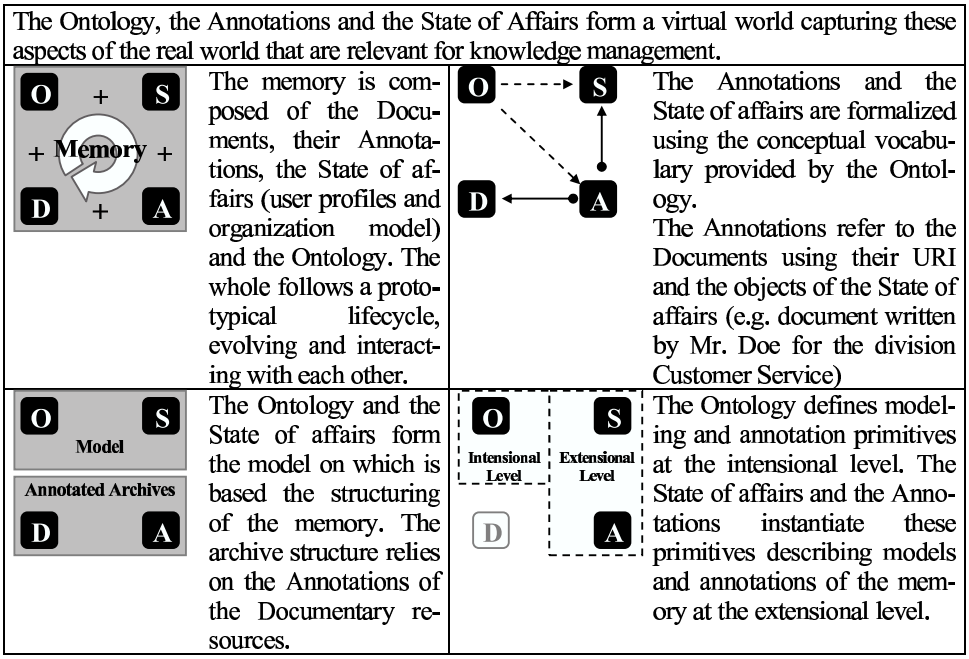

Figure 1. Four main elements of a corporate semantic web

We can summarize our approach in three stages:

- To apply scenario-driven knowledge engineering techniques in order to capture the needed conceptual vocabulary. We then specify the corporate memory concepts and their relationships in an ontology and we formalize them in RDFS or OWL.
- To use the conceptual vocabulary of the ontology and the scenario analysis to develop corporate and user models. These models are implemented in RDF and instantiate the RDFS/OWL ontology description.
- To structure the corporate memory using RDF annotations on the documents: these annotations instantiate the RDFS/OWL ontology description and make reference to the corporate and user models.

Among the domain applications where the ACACIA team implemented corporate semantic webs and used CORESE are:

- SAMOVAR: a system supporting a memory of vehicle projects for the car manufacturer Renault [20], and answering queries such as: “Find all fixing problems that occurred on the dashboard in a past project”.
- CoMMA: a multi-agent system for corporate memory management supporting the integration of a new employee and technological watch [18]. It answers distributed queries over distributed annotation bases such as “Find users who are interested in the technological news that was submitted about GSM v3”.
- KMP: a public knowledge management portal to cartography skills of firms in the Telecom Valley of Sophia Antipolis [26]. It answers queries such as: “Who are the possible industrial partners knowing how to design integrated circuits within the GSM field for cellular/mobile phone manufacturers?”.
- Life-line: a virtual staff for a health network [13] that guides physicians discussing the possible diagnoses and the alternative therapies for a given pathology, according to the patient’s features.
- MEAT: a memory of experiments of biologists on DNA microarray relying on automated annotation of scientific articles [23]. It answers queries such as "Find all the articles asserting that the HGF gene plays a role in a lung disease".

4 Semantic web services

4.1 Overview on web services

Web services, sometime called application services, are a standardized way of integrating applications over the Web. They rely on a collection of open standards used for exchanging XML-formatted data between applications or systems over Internet protocols.

> "*A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*"[4]

A Web service may be defined as a task-oriented, coarse-grained, XML-based online business service that is widely-accessible and that may be composite. A Web service provides a fixed function interface that may be invoked by any client; it is usually described through metadata for service consumers (name, description, version, QoS, etc.) Unlike traditional Web client/server models, Web services do not provide the user with a GUI but they offer a programmatic interface to share business logic, data and processes across a network. Thus, typically, a call to a Web service will be embedded in an application offering the GUI to interact with the users. Used primarily in B2B as a means for businesses to communicate with each other, Web services allow organizations to connect their IT systems with regard to a set of focused interactions and without intimate knowledge of each other's systems. Thanks to standardized data format and protocols, Web services are not tied to any programming language, operating system or platform and allow heterogeneous applications from different sources to communicate with each other.

The most prominent solution adopted to date in the industry to locate, describe, and invoke Web services is the trio of standards SOAP, WSDL and UDDI depicted in Figure 2.

- **SOAP** [47]: Since a broad range of applications will eventually be interconnected through the Web, the XML Protocol Working Group of the W3C is in charge to create simple protocols that can be ubiquitously deployed and easily programmed through scripting languages, XML tools, interactive Web development tools, etc. The goal is a layered system which will directly meet the needs of applications with simple interfaces, and which can be incrementally extended to provide the security, scalability, and robustness required for more complex application interfaces. They focus on the SOAP protocol, a simple XML-based messaging and remote procedure call (RPC) systems, layered on standard Web transport protocols such as HTTP and SMTP.
- **WSDL** [48]: One of the requirements for the development of Web services is the ability to describe the interface, i.e., the boundary across which applications (Web services user agents and Web services) communicate. The Web Services Description Working Group of W3C is chartered to design the following components of the interface: (i) the message, a definition for the types and structures of the data being exchanged; (ii) the message exchange patterns, the descriptions of the sequence of operations supported by a Web service; and (iii) the protocol binding, a mechanism for binding a protocol used by a Web service, independently of its message exchange patterns and its messages.
- **UDDI** [33]: The Universal Description Discovery & Integration (UDDI) is a definition of a set of services and markup languages supporting the description and discovery of (i) organizations, and in general Web services providers, (ii) the Web services they make available, and (iii) the technical interfaces which may be used to access those services. Based on a common set of industry standards, including HTTP, XML, XML Schema, and SOAP, UDDI aims at offering an interoperable, foundational infrastruc-

ture for a Web-service-based software environment for both publicly available services and services only exposed internally within an organization.

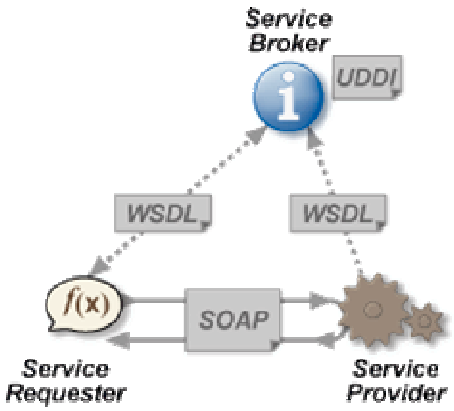

Figure 2. Service-oriented architecture [52]

A number of extensions have been proposed to address some shortcomings of the classic Web services technologies:

- **WSMF** [41]: The Web Services Management Framework is a logical architecture for the management of resources, including Web services themselves, through Web services. This framework is based on the notion of managed objects and their relationships: a managed object essentially represents a resource and exposes a set of management interfaces through which the underlying resource could be managed. Similarly, relationships among managed objects represent relationships among underlying resources.
- **OWL-S** [34]: Early efforts were made to define Web Service ontologies and markup such as DAML-S and its successor OWL-S. To make use of a Web service, software needs a computer-interpretable description of the service. Relying on semantic Web frameworks, service providers could be provided with a set of basic classes and properties for declaring and describing services. The OWL-S initiative of the American DARPA/DAML project aims at providing such an OWL-based Web Service Ontology, as well as supporting semantic Web tools to enable: automatic Web service discovery, automatic Web service invocation, automatic Web service composition and interoperation, and automatic Web service execution monitoring.
- **BPEL4WS** [2]: The Business Process Execution Language for Web Services provides a language for the formal specification of business processes and business interaction protocols. Processes in BPEL4WS export and import functionality by using Web service interfaces exclusively. By doing so, BPEL4WS extends the Web services interaction model and enables it to support business transactions. BPEL4WS defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and the business-to-business spaces.
- **BPML** [3]: The Business Process Modeling Language (BPML) is a meta-language for modeling business processes, just as XML is a meta-language for modeling business data. BPML offers an abstracted execution model for collaborative and transactional business processes based on the concept of a transactional finite-state machine. BPML considers e-business processes as made of a common public interface and as many private implementations as process participants. This enables the public interface of BPML processes to be described independently of their private implementations.

- **WSCI** [49]: The Web Service Choreography Interface is an XML-based interface description language that describes the flow of messages exchanged by a Web service participating in choreographed interactions with other services. WSCI describes the dynamic interface of the Web service participating in a given message exchange by means of reusing the operations defined for a static interface. WSCI works in conjunction with WSDL. WSCI describes the observable behavior of a Web service by means of a message-flow oriented interface. This is expressed in terms of temporal and logical dependencies among the exchanged messages, featuring sequencing rules, correlation, exception handling, and transactions. WSCI also describes the collective message exchange among interacting Web Services, thus providing a global, message-oriented view of the interactions.
- **SWWS** [41]: The IST project "Semantic Web enabled Web Services" aims at overcoming the initial limitations of UDDI for service discovery by relying on semantic Web technologies to: (i) offer a comprehensive Web service description framework; (ii) define a Web service discovery framework; and (iii) offer a scalable Web service mediation middleware.
- **XLANG** [55]: This is a notation for the specification of message exchange behaviors among participating Web services. XLANG is expected to serve as the basis for automated protocol engines that can track the state of process instances and help enforce protocol correctness in message flows.
- **WSCL** [50]: The Web Services Conversation Language allows the abstract interfaces of Web services, i.e., the business level conversations or public processes supported by a Web service, to be defined. WSCL specifies the XML documents being exchanged, and the allowed sequencing of these document exchanges. WSCL conversation definitions are themselves XML documents and can therefore be interpreted by Web services infrastructures and development tools. WSCL may be used in conjunction with other service description languages like WSDL, e.g., to provide protocol binding information for abstract interfaces, or to specify the abstract interfaces supported by a concrete service.
- **WSFL** [51]: The Web Services Flow Language is an XML language for the description of Web services compositions. WSFL considers two types of Web services compositions: (i) the first type specifies the appropriate usage pattern of a collection of Web services, in such a way that the resulting composition describes how to achieve a particular business goal (typically, the result is a description of a business process); (ii) the second type specifies the interaction pattern of a collection of Web services (in this case, the result is a description of the overall partner interactions).

One of the assets of web services is that they not only provide a standard way to advertise and invoke business services over the web, they also aim at enabling the composition of these services available online to provide complex services with high added-value. Literature splits the problem of coordination of Web services into several sub problems: (i) service discovery, (ii) dynamic service composition planning, and (iii) service execution monitoring. Up to now there is no commonly agreed technique to compose services and monitor the execution of such a composition.

4.2 Positioning w.r.t. semantic web services

Web services standards (WSDL, UDDI, SOAP) [11] reach their limits when we want to automate some tasks like discovery, invocation or composition. So many approaches, mainly relying on Semantic Web technologies, have been proposed recently to overcome these limits mainly due by the absence of semantics in the current web services description and registration technologies.

Semantics Web Services (SWS) are services which are described using enriched markup languages and able to be automatically discovered, executed and composed. Semantic Web Ser-

vices frameworks allow service developers to enrich the service descriptions with formal annotations of their capabilities. These semantic descriptions aim at enabling applications to discover, to invoke and to compose the annotated services in an automatic manner [28]. Many frameworks have been proposed [6] among which the main ones are OWL-S, WSMO, WSDL-S and IRS.

4.2.1 OWL-S

OWL-S [29] [34] is a set of ontologies for describing web services. It has been developed to provide the building blocks for encoding rich semantic service descriptions based upon OWL the semantic web language recommended by W3C. It consists of three main upper ontologies used to describe three facets of the services:

- The *Profile* facet is essentially used for describing the non-functional properties (service name, category, quality of service, etc.);
- The *Process* facet gives a detailed description of a service operation, its inputs and outputs and can even detail its internal processes and, if it is the case, it identifies the other services it is composed of;
- The *Grounding* facet provides details on how to interoperate with a service via messages.

The service profile gives the information needed for an application to discover a service. The service model and service grounding offer the information needed for an application to make use of a service.

In [45], the authors show how OWL-S can be used with UDDI to perform a discovery mechanism based on semantic search. They propose OWL-S/UDDI matchmaker which takes advantage of proliferation UDDI registries in the web services technology infrastructure and of the explicit capability representation of OWL-S. OWL-S profile descriptions are stored inside an UDDI registry and can be processed with an OWL-S matchmaker module associated to the UDDI registry. A mapping technology is also provided to transform OWL-S profile into the UDDI data model. The matchmaking algorithm used in this approach defines a flexible mechanism based on the subsomption mechanism of OWL. When a request is submitted, the algorithm finds an appropriate service by first matching the output of the request against the outputs of the published advertisements, and then, if any advertisement is matched after the output phase, the inputs of the request are matched against the inputs of the advertisements matched during the output phase. The degree of match between two outputs or two inputs depends on the match between the concepts they represent.

4.2.2 SWWS

Semantic Web enabled Web Services (SWWS) [5] is a project [41] which aims at providing a web service description framework, a web service discovery framework and a mediation platform for web services. This work relies on a conceptual architecture. A result of the SWWS project is the Web Service Modeling Framework (WSMF) [29]. WSMF provides a conceptual model for developing and describing web services and their composition. It consists of four main elements: ontologies that provide the terminology used by other elements, goal repositories that define the problems that should be solved by web services, web services definitions that define various aspects of a web service, and mediators for interoperability problems [6].

The Web Service Modeling Ontology (WSMO) [54] is an ontology for describing various aspects related to Semantic Web Services following WSMF. To describe semantically Web Services, WSMO proposes two kinds of attributes: a capability which is a non functional description of a Web Service (preconditions, post-conditions, assumptions, effects), and service interfaces which specify the behavior of the service to achieve its functionality by offering information about the operational competence on the web service (how a client can communicate with the service, how the overall service functionality can be achieved from other services).

WSMO relies on a proprietary language defined in the SWWS project, WSML (Web Service Modeling Language) [53]. WSML uses logical formalisms to enable the description of various aspects related to Semantic Web Services.

The reference implementation of WSMO is WSMX (Web Service Execution Environment) [22] which provides dynamic discovery, mediation, selection and invocation.

4.2.3 WSDL-S

While OWL-S and SWWS approaches are essentially top-down approaches (i.e. started at the knowledge level and then grounded in the WSDL) the WSDL-S approach [1] considered starting from WSDL and augmenting its expressivity with semantic descriptions. Using extension slots of WSDL, WSDL-S provides a mechanism to add annotations in a WSDL description in order to describe semantically the capabilities and requirements of Web services (inputs, outputs, preconditions, effects, operations). Again, these annotations are based on formal descriptions of concept types provided by external ontologies.

This approach is a refinement of the original WSDL-S proposed in the METEOR-S (Managing-End-To-End-OpeRations for Semantic Web Services) project [30].

4.2.4 IRS

IRS-II (Internet Reasoning Service) [32] is a SWS framework and implemented infrastructure which allows service developers to semantically describe and execute web services. It builds on past results in knowledge modeling and in particular on the UPML framework [17] which allows developers to separate services from problem specification. This framework partitions knowledge into ontologies, domain models, task models, and problem solving methods which are connected via bridges. The main goals of IRS-II are to support the publication, location, composition and execution of heterogeneous web services augmented with semantic description of their functionalities. Descriptions are in OCML (Operational Conceptual Modeling Language).

IRS-III [14] is a platform and infrastructure for creating WSMO-based Semantic Web Services, built upon the IRS-II implementation.

4.2.5 WSDF

The Web Service Description Framework (WSDF) [15] is a suite of tools providing Web services with semantic annotations allowing the ad-hoc invocation of a service without any prior knowledge of the API. The sole prerequisite is a shared ontology defining major domain concepts; WSDF requires client and server to offer a mapping from the local structures to a common domain ontology at design time. WSDF can be applied to clients and services written in a conventional object oriented programming language.

4.2.6 Our position in the SWS stack

Because OWL-S is directly expressed in OWL and because our approach to corporate semantic webs relies on OWL too, we relied on OWL-S for our work. However, in our current scenarios, we only use the profile and the grounding of OWL-S plus the input and output description in the process description. This corresponds in WSMO to the services capabilities and input output description and also to the semantics added by WSDL-S to annotate services. Figure 3 approximates the double stack of semantic web services and the grey area symbolizes the parts we implemented: XML 1.0, WSDL 1.1 et SOAP 1.2 (JWSDP 1.3), RDF(S) 1.0, OWL-S 1.1.

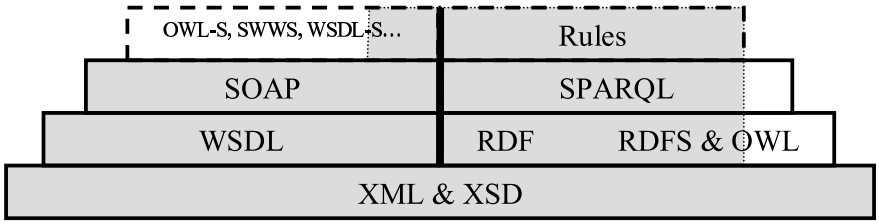

Figure 3. Implemented parts in the semantic web services stacks

4.3 Composing semantic web services

One of the main objectives of SWS frameworks is to provide automatic composition of services. Several surveys have been done about web service composition issue [46][31][8][39].

One can note that, in the majority of SWS frameworks, the proposed composition approach is just an interactive one. In such composition methods, users hold the control of the definition and are assisted by the software for the discovery and filtering of candidate web services.

[43] proposes an interactive composition approach using matchmaking algorithms to assist users to filter and select services during composition process. Web services are described with OWL-S annotations which are used to dynamically build up a filtering panel where constraints about some properties of the service may be entered. But, the composition process must be guided by a human with domain knowledge for the task. The presented framework just helps the user in a semi-automatic manner by offering relevant choices at each step; then, the user can makes selection.

[40] presents an interactive composition approach based on the IRS-III framework. A graphical Java tool has been developed to support users for the definitions of dynamic composition by recommending goals according to the context at each step of a composition. Recommendations are done by matching the inputs and outputs of the goal that were previously selected considering also the subsumption links of the input and output types. During the process composition, the user can also select mediators to map and perform transformations between goals; he is also able to select an "If-Then-Else" control flow operator. A Java API is provided for orchestration by instantiation of the service components and control operators defined in the composition according to their data dependencies. The API enables to build, validate and write a composite service to IRS server.

The framework presented in [25] also allows interactive composition of services. The system assists users to construct a computational pathway by using semantic description of services; the approach uses an analysis tool that helps users create compositions of web services. A computational way consists of "a set of operations and a set of links that connects the operations based on their input and output parameter constraints". The framework has been applied to compose computational pathways in earthquake science where engineers interactively compose web services in order to answer their queries (i.e. analyze the hazard level of a given site). Task and domain ontologies represented in description logic are used to add semantics on WSDL descriptions of services, allowing to analyze a sketch of a composition of services based on the definitions of task types and their input / output data types, and to generate error messages and specific suggestions to users. The developed tool offers a text editor where users can select services and make links by clicking their input and output parameters.

[21] describes a framework to design and compose SWS. The presented framework is based on a set of ontologies that describe SWS at the conceptual (i.e. knowledge) level by making explicit its features (access, descriptive, functional and structural) independently of a language. The ontology set is composed of: an ontology describing the upper-level concepts that define the features of a SWS based on OWL-S, an ontology of problem-solving methods based on the UPML specification, an ontology used to describe the primitives of the knowledge representa-

tion model in which the domain ontology is described. This last one is built on the top of another ontology based on the XML Schema Datatypes and used to describe the types of the concepts and attributes. These ontologies can be instantiated to design and compose manually (or in semi-automatic way) SWS at a knowledge level. The implemented prototype provides a graphical interface where users can design and compose SWS at a conceptual level.

One of the rare ones to propose an automatic approach is Peer [38] that presents a tool, named WSPlan, for automatic web service composition. The approach is based on the transformation of web service composition problems into AI planning problems (expressed by PDDL specifications), and the dynamic choice of the most suitable planner for a particular task. A proprietary lightweight service description format is used to describe the web service semantics; this annotation format is used as a bridge between WSDL web service descriptions and PDDL specifications. WSPlan transforms web service annotation data and information to PDDL documents. Then, a planner able to process PDDL that fits the requirements imposed by the particular domain definitions is used to identify a plan. The resulted plan is transformed into a flow of web service operations to be invoked.

In our scenarios, we do need to offer high-level functionality through dynamic integration. However we have not found ergonomic ways to describe and decompose service needs to support fully automatic composition. In addition, such a functionality seems to rely a lot on domain knowledge, and we think that, as claimed in [25], in many contexts users will want to control the composition process, influencing the service selection. We found more realistic to consider for instance the request from business managers to be able to implement business workflows in flexible (declarative) manners above the classical web services architectures. So, in our experiment, we exploit some capabilities of CORESE to provide interactive service composition by assisting users during the composition process. Also, by composing CORESE and corporate web services, we are able to compose services with the knowledge of the corporate semantic web.

5 Corporate semantic web services: discovering corporate application

5.1 Semantic annotation of corporate applications

As explained in the previous section, since our approach to corporate semantic webs relies on OWL we relied on OWL-S for our annotations of the web services; OWL-S offers the framework the closest to semantic web frameworks and thus is directly compatible with CORESE. WSML or WSDL-S would have required mappings. We use the profile, the process part offering input and output descriptions and the grounding of OWL-S to annotate web services wrapping corporate applications.

In our current prototype, to wrap a corporate application into an annotated service, one must (i) write and deploy the corresponding web service; and (ii) annotate the web service with OWL-S.

The OWL-S code below shows the annotation of a service based on an INRIA intranet LDAP application allowing employees to get the phone number of a secretary of another employee.

```
<service:Service rdf:ID="PosteService_Secretaire">
   <service:presents rdf:resource="#Profile_Poste_Service_Secretaire"/>
   <service:describedBy rdf:resource="#PosteSecretaire"/>
   <service:supports rdf:resource="#PosteGrounding_Secretaire"/>
</service:Service>
<profile:Profile rdf:ID="Profile_Poste_Service_Secretaire">
   <service:presentedBy rdf:resource="#PosteService_Secretaire"/>
   <profile:has_process rdf:resource="#PosteSecretaire"/>
   <profile:serviceName>PosteSecretaire</profile:serviceName>
   <profile:textDescription>INRIA LDAP</profile:textDescription>
   <profile:hasInput rdf:resource="#PosteSecr_input"/>
   <profile:hasOutput rdf:resource="#PosteSecr_output"/>
</profile:Profile>
<process:AtomicProcess rdf:ID="PosteSecretaire">
   <service:describes rdf:resource="#PosteService_Secretaire"/>
   <process:hasInput>
      <process:Input rdf:ID="PosteSecr_input">
       <process:parameterType>&xsd;#string</process:parameterType>
       <process:semanticType rdf:resource="&doc;#EmployeeName"/>
      </process:Input>
   </process:hasInput>
   <process:hasOutput>
       <process:Output rdf:ID="PosteSecr_output">
         <process:parameterType>&xsd;#string</process:parameterType>
         <process:semanticType rdf:resource="&doc;#AssistantPhone"/>
       </process:Output>
     </process:hasOutput>
</process:AtomicProcess>
<grounding:WsdlAtomicProcessGrounding rdf:ID="PosteSecrGrounding">
  <grounding:owlsProcess rdf:resource="#PosteSecretaire"/>
  (...)
  <grounding:webserviceURL>
    <xsd:anyURI rdf:value="http://localhost:8080/jaxrpc-Poste/poste"/>
  </grounding:webserviceURL>
</grounding:WsdlAtomicProcessGrounding>
(...)
```

Figure 4: Example of service description with OWL-S

In bold, we emphasized our small extension of the OWL-S Parameter concept: we added a property named *semanticType* which provides us with the possibility to integrate services with the corporate memory as we shall see in section 6. In this example, the given service gets an employee name as input and provides an assistant phone as output.

The part in italic shows a small extension (*webserviceURL* attribute) to the grounding OWL-S ontology to have the ability to store the service SOAP address. This information

(SOAP address) is the only one we need in our experiments; it is stored in the service WSDL interface, and not present in the OWL-S part we use.

In the experiment presented in this report, all our services are based on JWSDP 1.3 for their implementation and wrap services of our intranet.

5.2 CORESE as a corporate semantic UDDI registry

As shown in Figure 5, internally CORESE relies on a mapping to conceptual graphs and thus leverages results of more than 20 years of research and implementation in that branch of knowledge-based systems and knowledge representation.

In the corporate memories developed so far, the annotations generally describe documentary resources or corporate structures, but, when relying on schemata as the ones surveyed in section 4.2, these annotations can describe web services available online (intranet, extranet, Internet). This means that CORESE allows us to automate the identification of web services available to a user. Following a service-oriented architecture and a *find-bind-execute* schema [35] CORESE fits well in the picture with semantic web services as a semantic UDDI registry (Figure 5):

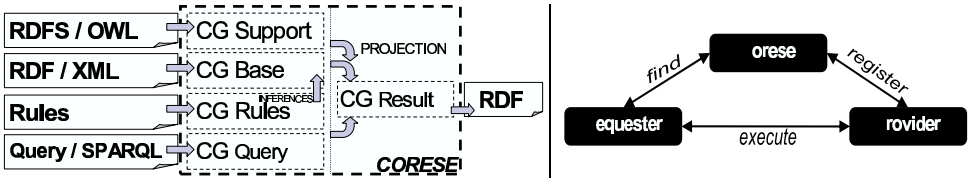

Figure 5. CORESE as a semantic UDDI registry

In this new architecture, we moved from text-based UDDI search to the semantic search engine CORESE to solve queries on the descriptions of the services, taking into account the ontologies used to characterize them and leveraging their semantics when solving query. With this architecture, annotations of services corresponding to corporate applications are stored in the corporate semantic web. Then, the services can be automatically discovered and dynamically invoked without any prior knowledge of their descriptions.

5.3 Semantic Web portal to corporate services

Our current implementation (Figure 6) is embedded in the CORESE semantic web server architecture.

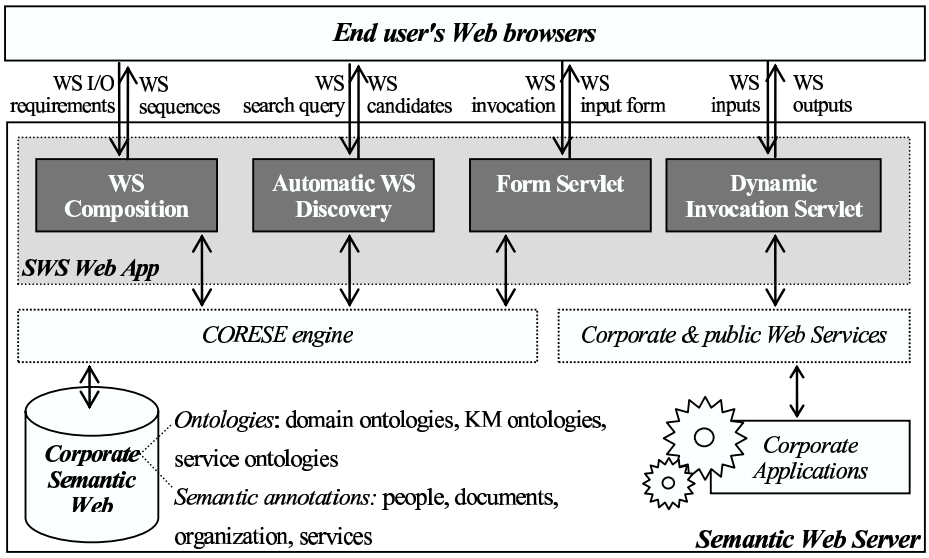

Figure 6. Architecture of the portal

The CORESE semantic web server is developed in Java according to a 3-tier architecture. We added to this architecture a web application for semantic web services. The web application provides a semantic web portal to corporate (and external) services with three main functionalities:

1) *Automatic web service discovery*: it is achieved through Java Server Pages and XSLT stylesheets and provides automatic discovery of web services using CORESE queries upon their annotations just as for other resources of the corporate memory. Using the interface of the portal, services can be discovered in three ways (Figure 7):
   - by their category : selling service, information service, etc.;
   - by giving an available input and a desired output;
   - by traversing the list of atomic or composite services the annotations of which are saved in the memory.

   This requires IT managers or automatic crawlers to discover existing services and their annotations, and add them to the corporate memory.
2) *Dynamic invocation of web service*: it is realized by two Java servlets allowing (i) to get the necessary information about a service from the knowledge base of service descriptions in order to generate a form offering an interface to call the web service and (ii) to generate a dynamic client and the call to the web services.

To achieve the dynamic invocation of a selected atomic process, we perform the following tasks:

- get information about the inputs and outputs of the process from the Process part of the annotation : abstract name, data and semantic types ;
- for each input found in the Process, get its concrete name from the Grounding part of the annotation (the concrete name is the same used in the WSDL interface of the service);
- get the operation and port names, the URL of the service and its WSDL namespace from the Grounding;
- create a form to offer an interface for the inputs;
- create a dynamic client with the information got from the knowledge base and the inputs values provided by the end user. To generate the dynamic client, we use the Dynamic Invocation Interface (DII) API provided with JWSDP. The DII technique allows the possibility to access a previously unknown service.

3) *Web service composition*: it is achieved by some HTML pages and two Java servlets allowing to assist end-users to perform service composition : interactive composition (section 5.4), finding sequential compositions of services (section 5.5).

All the components of the portal rely on the CORESE engine (we use the CORESE API 2.1) to access service ontologies and annotations stored in the same knowledge base of the corporate semantic web. This presents the main advantage to enable us to perform composition between the services and the knowledge of the corporate memory (see section 6).

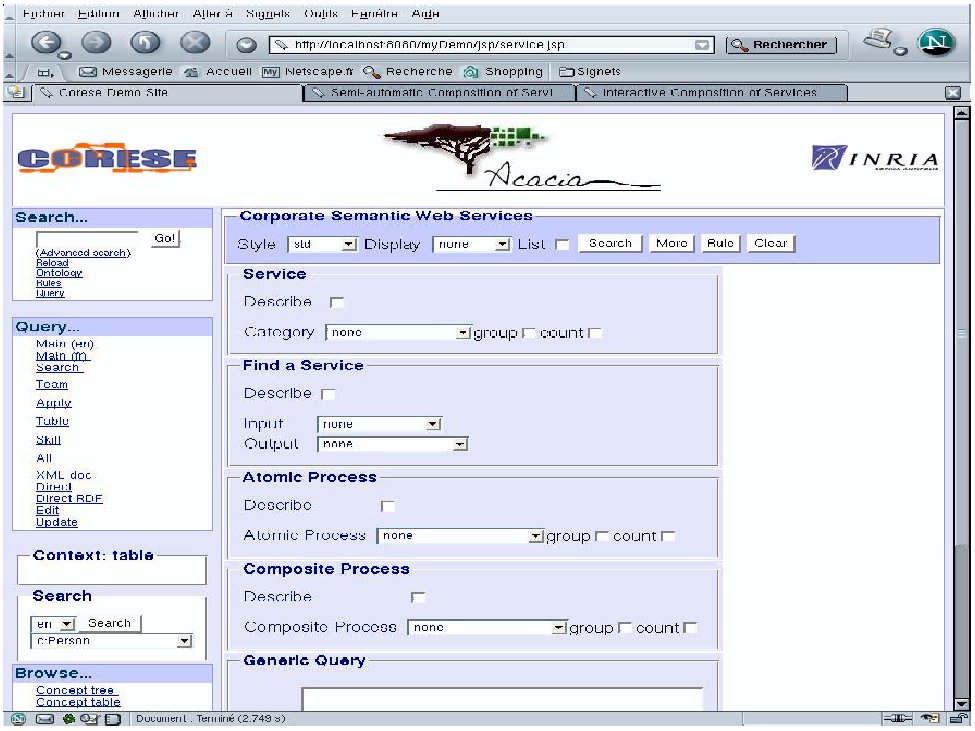

Figure 7: Semantic web portal to corporate services

As showed in the following example, when a service is selected by a user, instead of displaying the resource as it is the case for documents for instance, we dynamically generate a form thus offering an interface to call the service; on submitting the form, the inputs are used to generate a dynamic client and the call to the web services. The output is then simply displayed as a web page.

Figure 8 shows two windows:

- a window in the background showing the result of a query that retrieved a service description. This service is a mail-sender with a number of inputs. The user selected this service and obtained:
- the second window providing a form to specify the inputs. Once submitted, this form triggers a call to the web service which is then dynamically executed and provides a last window displaying the possible outputs in the web interface.

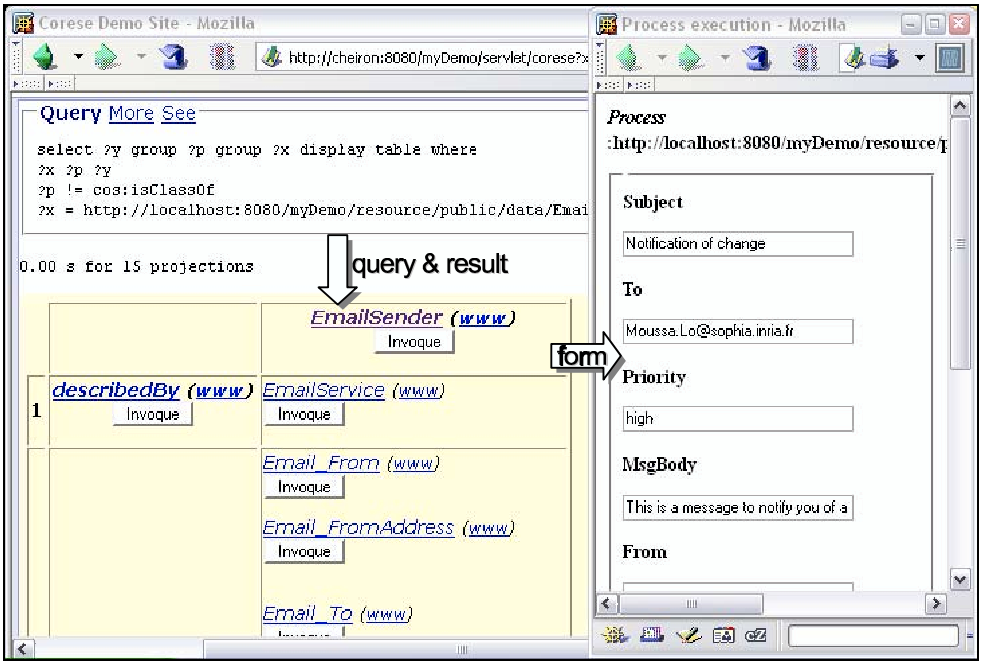

Figure 8: Discovering and Invoking a corporate web service with CORESE.

We also give the ability to export the output as an XML document. This can be interesting for instance to integrate the results in other applications. As we shall see later, it is the case of the CORESE service which provides an RDF/XML document as output to answer a query received as input.

5.4 CORESE-based interactive composition

When, according to a user need, no service is directly found from the portal, the user has the ability to perform a composition. We distinguish tow cases:

- The user has an available input and wants to find a list of services having this input as entry point;
- The user has a desired output and searches a list of services having this output as exit point.

For the first case the following steps help users to perform a service composition according to an available input:

1) The user provides an available input type, e.g. BookName;
2) The system proposes the list of services (atomic or composite process) able to receive this input type as entry (Figure 9a); this is done by performing CORESE queries about the semantic type attribute we have added to the OWL-S Parameter concept;
3) The user selects one of the proposed services. When at least two services are already selected during the composition process, the user can decide to stop the process and save his new composition in the form of an OWL-S composite process (Figure 9b and Figure 9c). In the current state of our prototype, we only offer the possibility to create a sequential composition of services.
4) If the user wants to continue and selects a service from the previous list, we search the services able to be composed with the previous selected service (Figure 9c). This is realized by matchmaking the semantic type attributes of the previous service and the other services stored in the knowledge base.

If no service is found, it is proposed to the user to create a composite service with the services he has already selected (Figure 9d).

Otherwise, the user is returned to step 3.

The composite services are saved in the service description knowledge base as OWL-S composite process and can discovered and invocated. The invocation of a composite service is done by invoking sequentially the processes it is composed of.

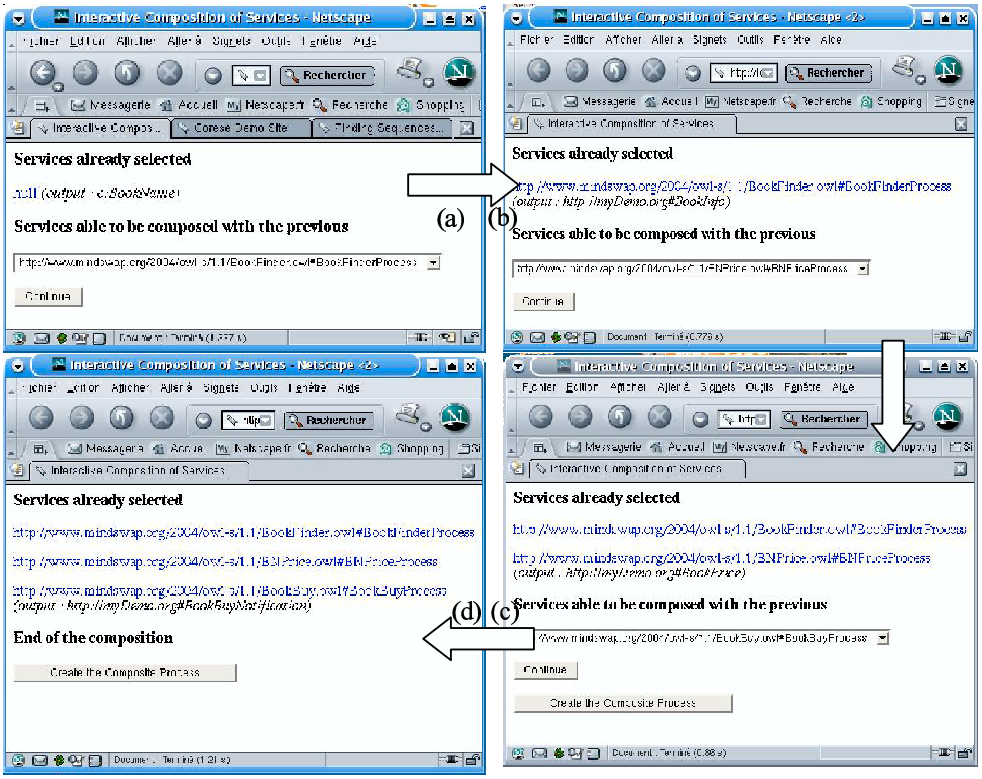

Figure 9. Examples of steps in an interactive composition (from an available input)

For the second case, in order to help users to perform a service composition according to a desired output, we provide almost the same steps as above:

1) The user provides a desired output type, e.g. BookBuyNotification;

2) The system proposes the list of services (atomic or composite process) able to provide this output type (Figure 10a);
3) The user selects one of the proposed services. When at least two services are already selected during the composition process, the user can decide to stop the process and save its composition in form of OWL-S composite process (Figure 10b and Figure 10c).
4) If the user wants to continue and selects a service from the previous list, we search the services able to be composed with the previous selected service (Figure 10c).

If no service is found, it is proposed to the user to create a composite service with the services he has already selected (Figure 10d).

Otherwise, the user is returned to step 3.

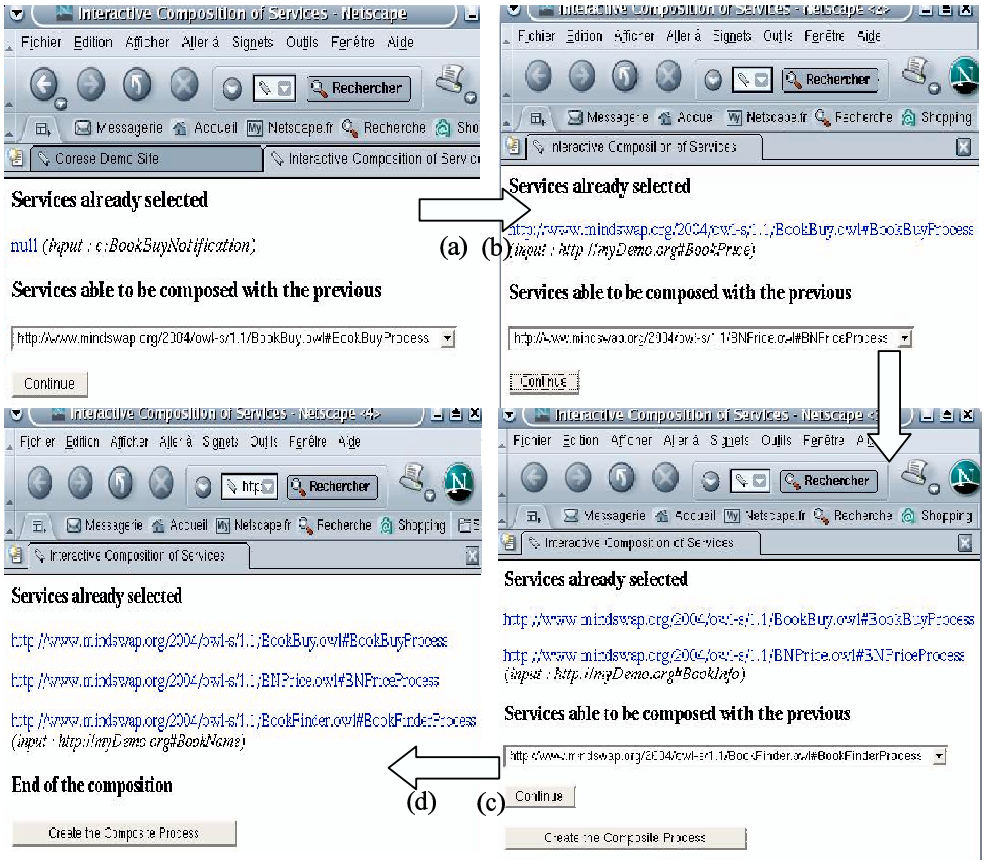

Figure 10. Examples of steps in an interactive composition (from a desired output)

5.5 Discovering sequential compositions of services using CORESE paths

We have started to introduce means to discover compositions of services that match a user's request expressed in terms of available inputs and desired outputs.

CORESE provides the possibility to search for resources linked by a path of (oriented or non-oriented) relations [10]. For instance `?x cos:Property[4] ?y` is a query that looks for an oriented path of a maximum length of 4 relations between two resources.

This feature of CORESE is usually used to explore the relations between two resources in the knowledge base (e.g.: to discover acquaintance networks). Applied to web services it can be used to discover a special type of composition: sequences *i.e.* a succession of services combined one after the other through their input and output types. The idea is to try to match the requirements of a user on available inputs and desired outputs by finding a sequence of services that link them.

Before doing so we had to formally define what it means for two services to be "composable" in a sequence. This is done through a production rule encoding the sufficient condition of the "composable" relation. To represent this relation, we perform another little extension of OWL-S by adding a property named "composable" for the Process concept. The rule defined in Figure 11 uses this new property and defines two services `s1` and `s2` as "composable" when the input of `s2` and the ouput of `s1` are ontologically compatible i.e. the type of the output of `s1` is the same or a subtype of the type of the input of `s2`.

```
<cos:rule>
    <cos:if>
        ?s1 rdf:type proc:Process
        ?s2 rdf:type proc:Process
        ?s1 proc:hasInput ?input
        ?s2 proc:hasOutput ?output
        ?s1 != ?s2
        ?input proc:semanticType ?inType
        ?output proc:semanticType ?outType
        ?outType rdfs:subPropertyOf ?inType
    </cos:if>
    <cos:then>
        ?s2 proc:composable ?s1
    </cos:then>
</cos:rule>
```

Figure 11. CORESE rule to formally define "composable" services

Applied to the knowledge base this rule generates the couples of "composable" services. Since this rule allows us to identify all the services that can be composed together we can then express queries over service descriptions to retrieve all sequences of services answering requests like "*Find all sequences of services having as input a BookName and as output a BookBuyNotification*" (Figure 12).

```
?s1 all::proc:composable[2] ?s2
?s1 proc:hasInput ?param1
?s2 proc:hasOutput ?param2
?param1 proc:semanticType c:BookName
?param2 proc:semanticType c:BookBuyNotification
```

Figure 12. CORESE query to find "composable" services

The Figure 13 and Figure 14 show an answer to this query with the Book services coming with the OWL-S API. In response to the previous query, we obtained the composition of the three services:
BookFinderProcess (input : *BookName*, output : *BookInfo*)

BNPriceProcess (input: *BookInfo*, output : *BookPrice*)

BookBuyProcess (input:*BookPrice*, output: *BookBuyNotification*)

Figure 13. CORESE answer to find "composable" services

We provide the ability to save sequences of services as OWL-S composite processes. These composite services can then be retrieved and executed like the other corporate web services. This feature can be used by an IT manager to create, save and propose new services from existing ones.

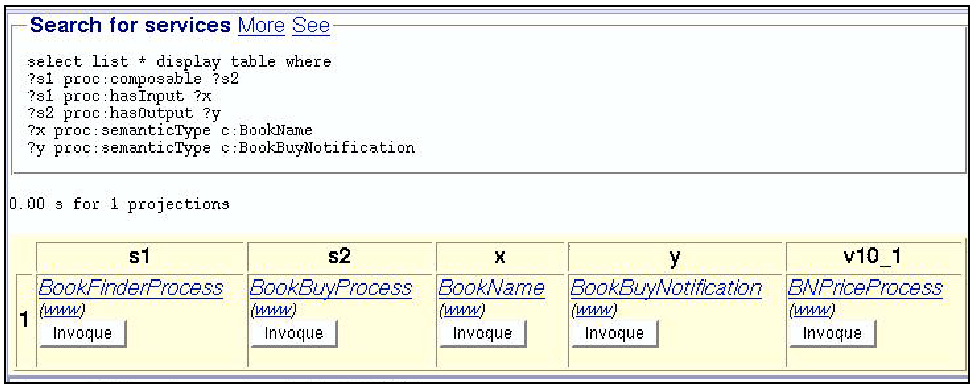

Figure 14. An example of a sequence of three services found by CORESE

6 Composing services with the knowledge of the corporate semantic web

In this section, we present how we use CORESE to demonstrate the composition between corporate services and the knowledge of the corporate memory. The first kind of composition is a mapping of services inputs types to CORESE queries and the second one is the use of CORESE as a semantic web service to access the corporate memory.

6.1 Mapping input types to queries

Since we are in a semantic web environment, "knowledge is everywhere" and the idea consists of using the knowledge stored in the corporate memory to populate, in an automatic way, the service inputs during execution. This idea was suggested by a previous work on context-aware service invocation [19]. The implementation was done in three steps:

- We associated to service inputs a predicate from a domain ontology by means of the *semanticType* attribute;
- We define these predicates using rules allowing to generate dynamically the needed information from the memory;
- When generating dynamically an invocation form from the grounding and the process to offer an interface to call the service, we also extract from the corporate memory the necessary information to populate the service inputs.

In the following example, we consider the service described in Figure 4 with an input associated to the domain ontology property *EmployeeName*. The CORESE rule defined in Figure 14 allows us to generate the candidate inputs from the corporate memory annotations; it defines a sufficient condition of the predicate *EmployeeName*.

```
<cos:rule>
    <cos:if>
       ?x rdf:type c:Employee
       ?x c:Name ?n
    </cos:if>
    <cos:then>
        ?x c:EmployeeName ?n
    </cos:then>
</cos:rule>
```

Figure 15. CORESE rule to define an input type

By using the generic CORESE query "*select ?value where ?x "+semanticType+"?value*" and by replacing *semanticType* by *c:EmployeeName,* we obtain from the memory the triples generated by the previous rule. So, as showed in the Figure 16, we can select a service from the result of a query on the directory, then we can pre-populate the input form (here we generate a dropdown box with the name of the employees for the service wrapping the LDAP application and described in Figure 4) and finally we display the result of the invocation. Again, every step of this process (semantic rules and queries) leverages the ontological reasoning.

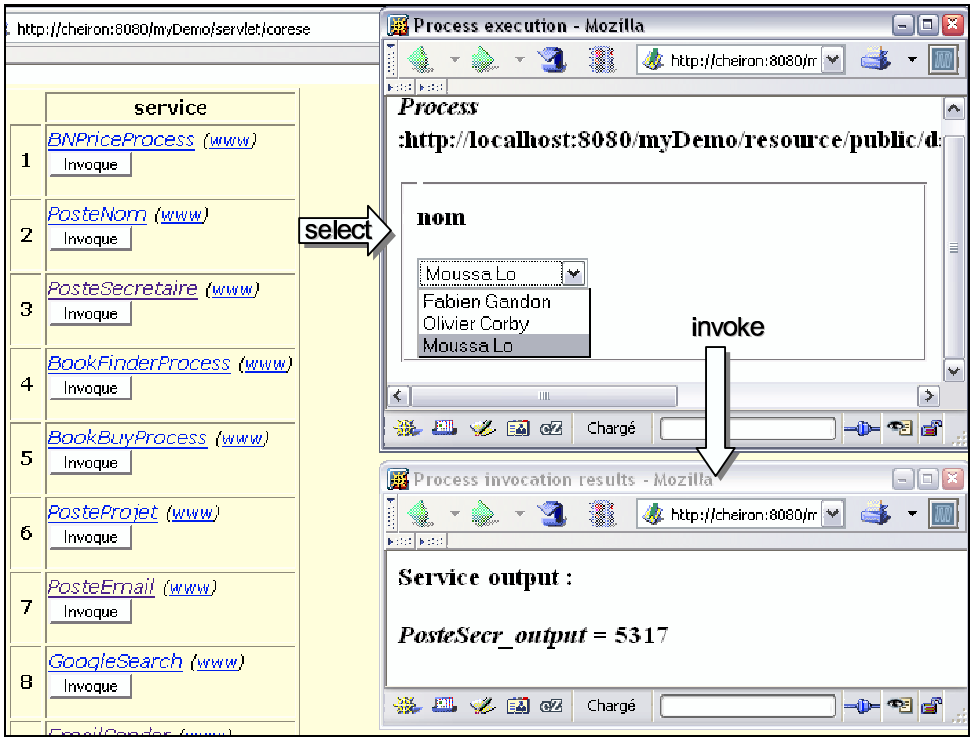

Figure 16. Leveraging the corporate memory to populate a SWS input

6.2 CORESE as a semantic web service to access to the corporate memory

The idea is to use semantic web services wrapping the CORESE engine in order to provide the possibility:

- to use the result of a query over the corporate memory as a service input;
- to use a service output to add knowledge to the memory.

In order to do so, we provide the ability to compose a service which wraps a corporate application with a CORESE semantic web service.

Two kinds of CORESE semantic web services can be used:

- a CORESE SWS which gets a query as input and gives the result as output; this service can be composed with any other one to get knowledge from the memory in order to associate a CORESE query and the service input (top part of Figure 17);
- a CORESE SWS which gets an RDF annotation as input, save and load the annotation into the memory; this service can be composed with another service to transform its output into knowledge for the memory (bottom part of Figure 17). In this case, we introduce an auxiliary XSLT service to transform a service output in a RDF annotation (the template of the annotation is given through an XSL stylesheet).

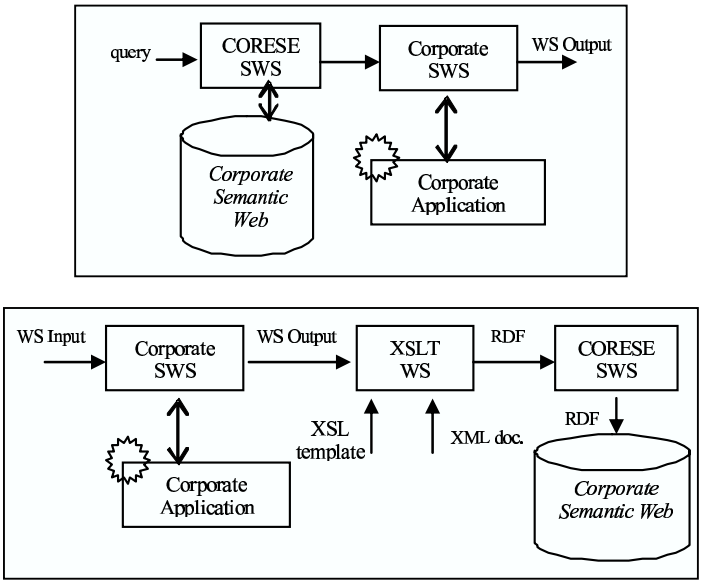

Figure 17. Connecting a corporate application and the corporate memory with CORESE SWS

We consider the following scenario. When someone wants to know the email address of an employee whose name he knows, he can use directly the service wrapping the LDAP application which gets an employee name and gives his email address as output. Now, assume he is searching the email address of the assistant of a given project (e.g. Acacia), but he doesn't know the name of this person. He could be able to perform a query (*Find the name of the Acacia project assistant*) over the memory and to give the result directly to the service. He could also add the new knowledge (email of the Acacia project assistant) to the memory for reuse.

The screenshot in Figure 18 shows the invocation of a service composed by:

- A CORESE Web service which gets a query as input, performs the query over the corporate semantic web and return the result; in this example, the query consists of "*finding the name of the Acacia project assistant" and the result is "Patricia Maleyran"*.
- A corporate service receiving an employee name (here the output of the previous service, i.e. *"Patricia Maleyran"*) and returns its email address.

In Figure 19, the screenshot shows the invocation of a service we can consider as the dual of the one presented above. This service is composed of:

- A service which receives an employee name and returns its email address;
- A service getting an RDF resource (here an Employee Name), an RDF object (here an email address), an RDF triple pattern and an empty XML document; it returns an RDF triple;
- A CORESE web service receiving an RDF triple, saves it as a new annotation and loads it into the memory.

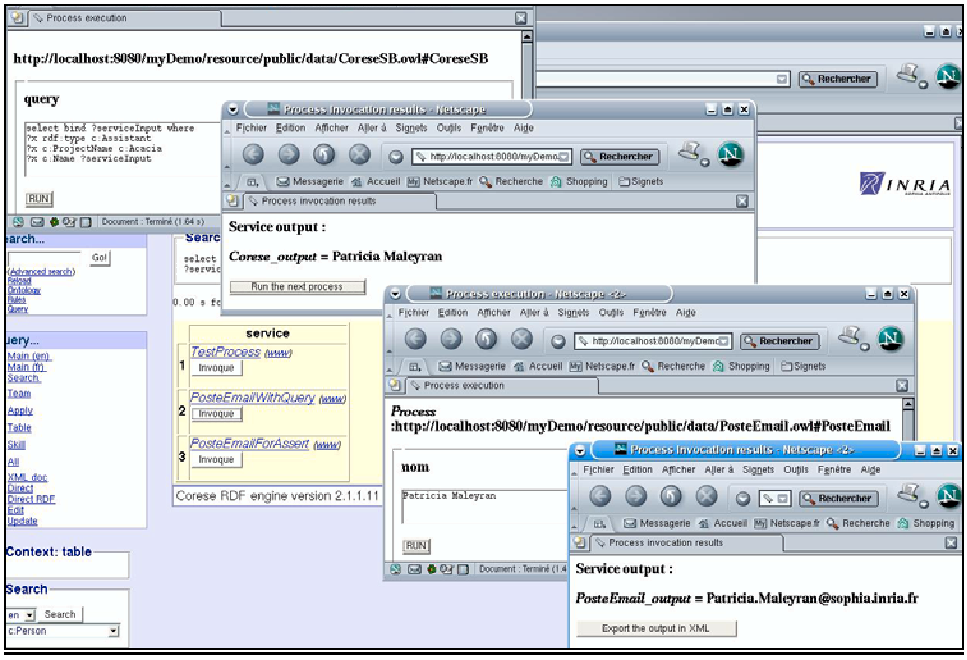

Figure 18. Example of connection from corporate memory to corporate application

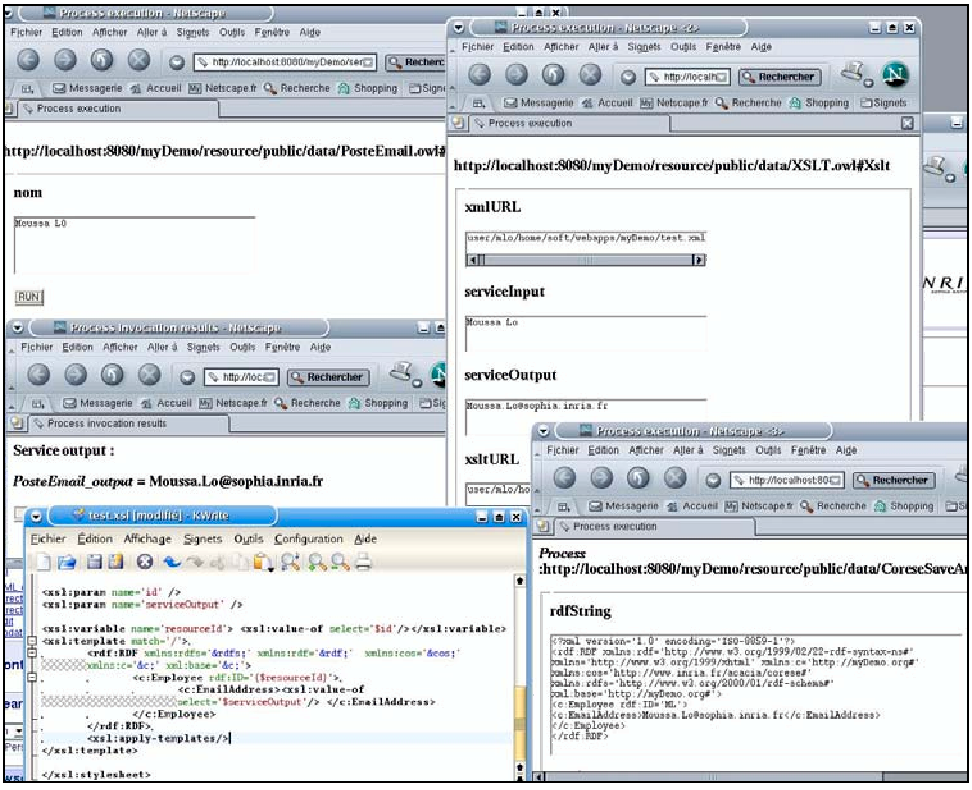

Figure 19. Example of connection from corporate application to corporate memory

7 Discussion and perspectives

In this report we presented a first experiment in integrating enterprise applications as web services in an intranet relying on semantic web and semantic web services frameworks. We chose to focus on a clearly identified family of scenarios: the integration of enterprise applications in a knowledge management system.

We have used CORESE, a semantic web search engine as a semantic UDDI registry. This allowed us to prototype a semantic web portal embedded in the CORESE semantic web server. The portal offers (i) automatic discovery, (ii) dynamic invocation, (iii) interactive composition and (iv) discovery of sequences of corporate and public web services.

The following improvements can be considered to enhance the current prototype:

- Improve the presentation of sequences of found services;
- Improve the creation of a composite service : for instance, provide the user with the ability to add annotations to new service;
- Improve the execution of a composite service: for instance, allow the possibility to replace an unavailable service by another.

The main contribution of this experiment is the composition of corporate web services with knowledge from the corporate memory:

- services inputs types are mapped into queries
- and the semantic search engine, CORESE, is used through a web service to connect corporate applications with the memory knowledge.

This experiment allowed us to differentiate between needed functionalities and prospective ones and to identify these layers of the semantic web service stack that we needed first.

A typical question, for instance, is the one of offering "manual vs. semi-automatic vs. fully automatic composition and invocation of services". In our scenarios, we do need to provide high-level functionality through dynamic integration. However we have not found ergonomic ways to describe and decompose service needs to support fully automatic composition. In addition, such a functionality seems to rely a lot on domain knowledge, and we think that, as claimed in [25], in many contexts users will want to control the composition process, influencing the service selection. We found it more realistic to consider for instance the request from business managers to be able to implement business workflows in flexible (declarative) manners above the classical web services architectures.

A second feedback we got from this experience is an urgent need for standardization and unification of the different contributions that could be involved in a complete solution. Indeed, these contributions are growing in number and complexity: WSDL, SOAP, OWL-S, WSMO & WSML, WSDL-S, WSFL, WSCI, WSCL, XLANG, BPEL4WS, SAML, XACML, etc. We are witnessing a multiplication of contributions for each and every stage of the life-cycles of web-services and especially discovery and binding [56][37][12][36][24] and discovery and composition [25][42][43][21]. There is clearly a need to homogenize the different approaches before the differences and incompatibilities (even the simply syntactic ones) hamper the foundations of Semantic Web Services such as interoperability.

Finally we are currently studying the interaction and integration with emerging semantic web extensions such as: SPARQL query language and protocol and SWRL rule description language. We also consider the problem of dynamically generating ergonomic user interfaces to semantic web services: web services are primarily designed for B2B programmatic interactions but the services or compositions of services are finally called by users through a client. Since their discovery, composition and invocation are dynamic this requires dynamically generated ergonomic user interfaces.

8 Acknowledgments

We thank AUF for partially supporting this work through the post-doctoral fellowship held by Moussa LO.

9 Bibliography

[1] Akkiraju, R., Farrell, J., Miller, J.A., Nagarajan, M., Schmidt M-T., Sheth, A., Verma, K. Web Service Semantics---WSDL-S, Technical Note, Version 1.0, April 2005, http://www.alphaworks.ibm.com/g/g.nsf/img/semanticsdocs/$file/wssemantic_annotation.pdf

[2] BEPL4WS, Business Process Execution Language for Web Services, http://www-130.ibm.com/developerworks/webservices/, 2002.

[3] BPML, Business Process Modeling Language, http://www.bpmi.org, 2004.

[4] Booth D., Haas H., McCabe F., Newcomer E., Champion M., Ferris C., Orchard D., Web Services Architecture, http://www.w3.org/2002/ws/arch/

[5] Bussler, C., Fensel, D., Maedche, A., A Conceptual Architecture for Semantic Web Enabled Web Services, ACM 2002.

[6] Cabral, L., Domingue, J., Motta, E., Payne, T., Hakimpour, F., Approaches to Semantic Web Services: An Overview and Comparisons, ESWS'04, 2004.

[7] Cao T., Dieng-Kuntz R., Fiès B., An Ontology-Guided Annotation System for Technology Monitoring, IADIS International WWW/Internet 2004 Conference, Madrid, Spain, 6-9 October 2004.

[8] Charif. Y., Sabouret N., An Overview of Semantic Web Services Composition Approaches, Electronic Notes in Theoretical Computer Science 85 N°. 6, Elsevier, 2005.

[9] Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., Querying the Semantic Web with the CORESE search engine. In Proc. of the 16th European Conference on Artificial Intelligence (ECAI'2004), Valencia, 22-27 August 2004, IOS Press, p. 705-709.

[10] Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., Gandon, F., Ontology-based Approximate Query Processing for Searching the Semantic Web with Corese, INRIA Research Report, July 2005.

[11] Curbera F. et al, Unraveling the Web Services: An Introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing, vol. 6, n° 2, 2002.

[12] Decker, K, Sycara, K, and Williamson, M. Matchmaking and Brokering, In proc. of the Second International Conference on Multi-Agent Systems (ICMAS-96), The AAAI Press, 1996.

[13] Dieng-Kuntz, R., Minier, D., Corby, F., Ruzicka, M., Corby, O., Alamarguy, L., Luong, P.-H. Medical Ontology and Virtual Staff for a Health Network, EKAW2004, 2004.

[14] Domingue, J., Cabral, L., Hakimpour, F., Sell, D., Motta, E., IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services, Workshop on WSMO Implementations (WIW 2004), 2004.

[15] Eberhart A., Ad-hoc Invocation of Semantic Web Services, IEEE International Conference on Web Services, July 2004.

[16] Fensel, D., Bussler, C., The Web Service Modeling Framework WSMF, Electronic Commerce: Research and Applications, Vol. 1., 2002

[17] Fensel, D., Motta, E., Structured Development of Problem Solving Methods, IEEE Transactions on Knowledge and Data Engineering, 13(6), pp. 913-932, 2001

[18] Gandon, F., Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web, PhD Thesis in Informatics, 7th of November 2002, INRIA and University of Nice - Sophia Antipolis

[19] Gandon, F. and Sadeh, N., Semantic Web Technologies to Reconcile Privacy and Context Awareness, Web Semantics Journal. Vol. 1, No. 3, 2004.

[20] Golebiowska, J., Dieng, R., Corby, O., Mousseau, Building and Exploiting Ontologies for an Automobile Project Memory, K-CAP, ACM Press, 52-59, 2001.

[21] Gomez-Perez, A., Gonzalez-Cabero, R., Lama, M., A Framework for Design and Composition of Semantic Web Services, First International Semantic Web Services Symposium, AAAI, March 2004.

[22] Haller A., Cimpian E., Mocan A., Oren E., Bussler C., WSMX - A Semantic Service-Oriented Architecture, In Proceedings of the International Conference on Web Service (ICWS 2005). Orlando, Florida, 2005.
[23] Khelif, K., Dieng-Kuntz., R., Ontology-Based Semantic Annotations for Biochip Domain, KMOM Workshop ECAI2004 , 2004.
[24] Kifer, M., Lara, R., Polleres, A., Zhao, C., Keller, U., Lausen, H., Fensel, D., A Logical Framework for Web Service Discovery, workshop Semantic Web Services: Preparing to Meet the World of Business Applications, at The Third International Semantic Web Conference, Hiroshima, 2004.
[25] Kim, J., Gil, Y., Towards Interactive Composition of Semantic Web Services, First International Semantic Web Services Symposium, AAAI, March 2004.
[26] KmP, http://www-sop.inria.fr/acacia/soft/kmp.html
[27] Linthicum D. S., Enterprise Application Integration, Addison-Wesley Information Technology Series 1999, ISBN: 0201615835
[28] McIlraith, S., Son T. C., Zeng H., Semantic Web Services, IEEE Intelligent Systems, 16(2):46-53, 2001.
[29] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K., Bringing Semantics to Web Services : the OWL-S Approach, SWSWPC'04, LNCS n° 3387, 2004.
[30] METEOR-S: Semantic Web Services and Processes, http://lsdis.cs.uga.edu/Projects/METEOR-S/.
[31] Milanovic N., Malek M., Current Solutions for Web Service Composition, IEEE Internet Computing, December 2004.
[32] Motta, E., Domingue, J., Cabral, L., Gaspari, M., IRS-II : A Framework and Infrastructure for Semantic Web Services, ISWC'03, 2003.
[33] OASIS, UDDI Specification, http://www.uddi.org
[34] OWL-S Coalition, OWL-S Specification, http://www.daml.org/services/owl-s/1.1/, 2004.
[35] Qusay H. M., Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI), April 2005, http://java.sun.com/developer/technicalArticles/WebServices/soa/
[36] Paolucci, M,, Kawamura, T., Payne, T. R, Sycara, K.; Semantic Matching of Web Services Capabilities, In Proc. of the International Semantic Web Conference (ISWC'02), Springer Verlag, Sardegna, Italy, June 2002.
[37] Paolucci, M., Soudry, J., Srinivasan, N., Sycara, K., A Broker for OWL-S Web Services, First International Semantic Web Services Symposium, AAAI, March 2004.
[38] Peer J., A PDDL Based Tool for Automatic Web Service Composition, Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR), September 2004, LNCS 3208.
[39] Rao J., Su X., A Survey of Automated Web Service Composition Methods, First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC), LNCS 3387, July 2004.
[40] Sell D., et al, Interactive Composition of WSMO-based Semantic Web Services in IRS-III, First AKT Workshop on Semantic Web Services (AKT-SWS), December 2004.
[41] Semantic Web Enabled Web Services Project, http://swws.semanticweb.org
[42] Sirin, E., Parsia, B., Planning for Semantic Web Services, Workshop Semantic Web Services: Preparing to Meet the World of Business Applications, at The Third International Semantic Web Conference, Hiroshima, 2004.
[43] Sirin, E., Parsia, B., Hendler, J., Composition-driven Filtering and Selection of Semantic Web Services, First International Semantic Web Services Symposium, AAAI, March 2004.
[44] W3C, SPARQL, http://www.w3.org/TR/rdf-sparql-query/

[45] Srinivasan N., Paolucci M., Sycara K., An Efficient Algorithm for OWL-S Based Semantic Search UDDI, SWSWPC'04, LNCS n° 3387, 2004.
[46] Srivastava. B., Koehler J., Web Service Composition – Current Solutions and Open Problems, ICAPS'03, 2003.
[47] W3C, SOAP Recommandation, http://www.w3.org/TR/SOAP
[48] W3C, WSDL Recommandation, http://www.w3.org/TR/WSDL
[49] W3C, WSCI, Web Service Choreography Interface (WSCI), http://www.w3.org/TR/wsci/, 2002
[50] W3C, WSCL, Web Services Conversation Language, http://www.w3.org/TR/2002/NOTE-wscl10-20020314/, 2002
[51] WSFL, Web Service Flow Language, http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, IBM, 2001.
[52] Web service, From Wikipedia, http://en.wikipedia.org/wiki/Web_Service, 11 of August 2005.
[53] WSML, Web Service Modeling Language, http://www.wsmo.org/wsml/
[54] WSMO working group, Web Service modeling Language, http://www.wsmo.org/2004/d2/
[55] XLANG, Web Services for Business Process Design, http://www.ebpml.org/xlang.htm, Microsoft, 2001.
[56] Zein, O. K., Kermarrec, Y., An Approach for Describing/Discovering Services and for Adapting Them to the Needs of Users in Distributed Systems, First International Semantic Web Services Symposium, AAAI, March 2004.